Introduction

- API

  An application programming interface (API) is a source code-based specification intended to be used as an interface by software components to communicate with each other.

- OWL API

  The OWL API is a Java API and reference implementation for creating, manipulating and serializing OWL Ontologies. The latest version of the API is focused towards OWL 2

  The OWL API is targeted primarily at representing OWL-DL.

  (latest: Download the latest binary release (3.2.4) - 22nd July 2011)

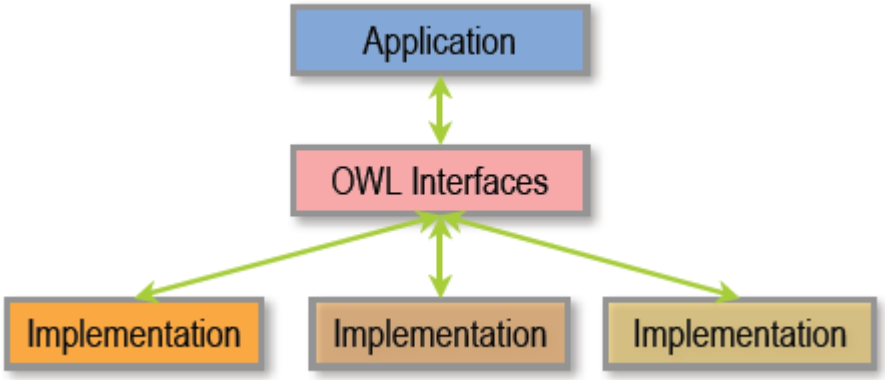
The OWL API includes the following components:

- An API for OWL 2 and an efficient in-memory reference implementation;
- RDF/XML parser and writer.
- OWL/XML parser and writer.
- OWL Functional Syntax parser and writer.
- Turtle parser and writer.
- KRSS parser.
- OBO Flat file format parser.
- Reasoner interfaces for working with reasoners such as FaCT++, HermiT, Pellet and Racer.

Why build an OWL API?

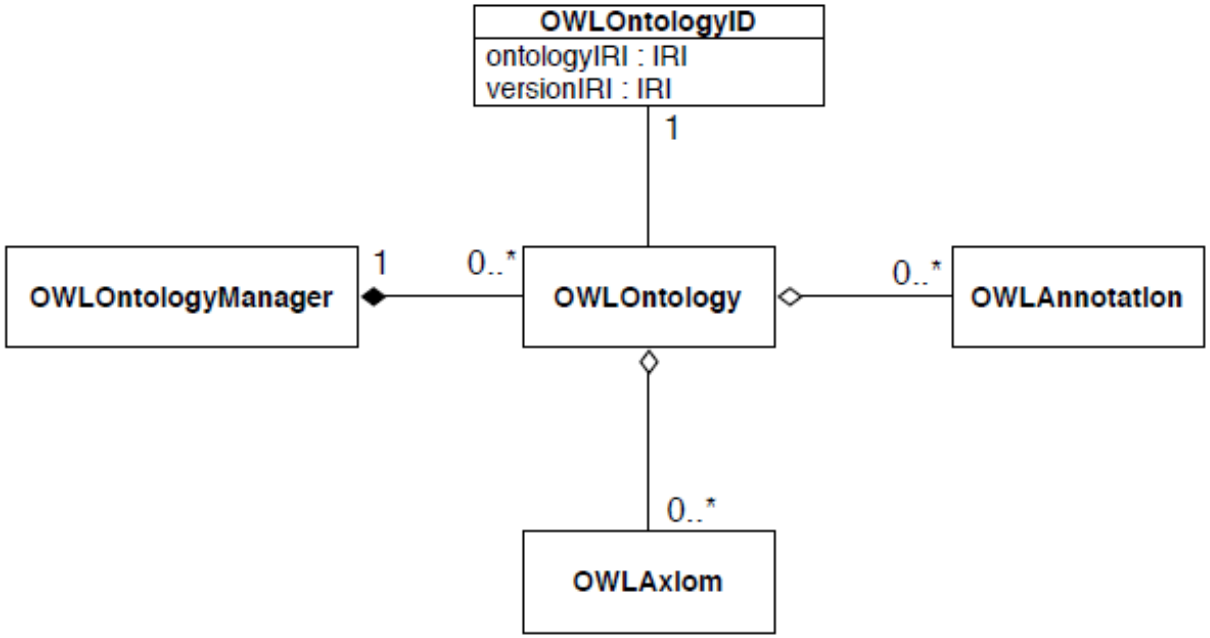
The use of a higher level data model can help to:

- insulate us from the vagaries of concrete syntax.
- make it clear what is happening in terms of functionality.
- increase the likelihood of interoperating applications.



OWL API and OWL 2.0

- The latest version of the OWL API has been designed to meet the needs [illegible] editors and C[illegible] d with the O[illegible]
- The l[illegible] ber of signifi[illegible] th the OWL [illegible] es has been [illegible] at was provic[illegible] model upon
- Unlike[illegible] l, the repres[illegible] evel of RDF triples. Indeed, the design of the OWL API is directly based on the OWL 2 Structural Specification



What to do with OWL API?

- A Code examples ng,
  m API
  s

  - Loading Ontologies - Shows how to load an an ontology
  - Saving Ontologies - Shows how to save an an ontology
  - Entities - Shows how to obtain references to entities (classes, properties, individuals etc.)
  - Data Ranges 1 - Shows how to work with data types and other data ranges
  - Data Ranges 2 - Shows how to work with user defined data ranges (e.g. int > 10)
  - Literals - Shows how to work with string, data values and language tags
  - Adding Axioms - Shows how to create an empty ontology, add axioms and save
  - Classes and Instances - Shows how to specify that an individual is an instance of a class
  - Property Assertions 1 - Shows how to specify that two individuals are related to each other.
  - Property Assertions 2 - Shows how to add an object property assertion (triple) to an ontology
  - Deleting Entities - Shows how to delete entities (classes, properties and individuals) from an ontology
  - Restrictions - Shows how to create restrictions and Â“add them to classesÂ” as superclasses
  - SWRL Rules - Shows how to create an ontology and add some rules
  - Reasoning - Shows how to interact with a reasoner
  - Visitors - Shows how to collect the properties that are used in restrictions on a given class
  - Annotations - Shows how to work with annotations such as labels and comments
  - Saving Inferred Axioms - Shows how to save inferred axioms into a new ontology, or back into an existing ontology
  - Merging Ontologies - Shows how two (or more) ontologies can be merged in a simple way
  - Walking Asserted Structure - Shows how to 'walk' over the asserted structure of an ontology.
  - Using Ontology IRI Mappers - Shows how to use OWLOntologyIRIMappers to redirect loading and loading of imports.
  - Module Extraction - Shows how to extract a locality based module from an ontology.

What to do with OWL API?

“OWLOntologyManager manager = OWLManager.createOWLOntologyManager();”

- Besides the model interfaces for representing entities, class expressions and axioms, it is necessary to have certain machinery that allow applications to manage ontologies.
- The OWLOntologyManager provides a central point for creating, loading, changing and saving ontologies, which are instances of the OWLOntology interface. Each ontology is created or loaded by an ontology manager. Each instance of an ontology is unique to a particular manager, and all changes to an ontology are applied via its manager.

What to do with OWL API?

Reasoner interface:

- The OWL API has various interfaces to support interacting with OWL reasoners. The main interface is the OWLReasoner interface which provides methods to perform the aforementioned tasks.
- The API allows a reasoner to be set up so that it listens for ontology changes and either immediately processes the changes or queues them in a buffer which can later be processed.
- The CEL, FaCT++, HermiT, Pellet, and Racer Pro reasoners provide OWL API wrappers. This means that they are also available as reasoner plugins to Protege-4.

```
36  public static void main(String[] args) throws OWLOntologyStorageException {
37      //
38      try{
39          OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
40          OWLDataFactory factory = manager.getOWLDataFactory();
41          IRI iri = IRI.create("file:///D:/Study/Softwares/StudySoftwares/Protege_3.4.8/HyperMusic.owl");
42
43          //load ontology Hypermusic owl
44          OWLOntology ont = manager.loadOntologyFromOntologyDocument(iri);
45          System.out.println("IndividualsInSignature"+ ont.getIndividualsInSignature());
46          System.out.println("ClassesInSignature"+ ont.getClassesInSignature()+"\n");
47
48          //save ontology Hypermusic
49          File file = new File("/C:/Users/Sichao Song/Desktop/HypermusicModified.owl");
50          manager.saveOntology(ont, IRI.create(file.toURI()));
51
52          //entities
53          OWLClass cls = factory.getOWLClass(iri);
54          PrefixManager pm = new DefaultPrefixManager("file:///D:/Study/Softwares/StudySoftwares/Protege_3.4.8/HyperMusic#");
55          OWLClass clsModified = factory.getOWLClass(":OntologyModified", pm);
56
57          OWLOntology ontology = manager.createOntology(IRI.create("file:///D:/Study/Softwares/StudySoftwares/Protege_3.4.8/Hy
58          OWLDeclarationAxiom declarationAxiom = factory.getOWLDeclarationAxiom(clsModified);
59          manager.addAxiom(ontology, declarationAxiom);
60          manager.saveOntology(ontology, new SystemOutDocumentTarget());
61          File file1 = new File("/C:/Users/Sichao Song/Desktop/HypermusicModified_1.owl");
62          manager.saveOntology(ont, IRI.create(file.toURI()));
63          manager.saveOntology(ontology, IRI.create(file1.toURI()));
```